CSED 490C

LAB 3

Jeongseop Yi (49004543)

Q1

in each cell, there are `MASK_SIZE * MASK_SIZE` floating point muliplication and addition, respectively. As there is `imageWidth * imageHeight` number of cells, there are `MASK_SIZE * MASK_SIZE * imageWidth * imageHeight` floating point muliplication and addition, respectively.

Q2

Let `w = TILE_WIDTH + Mask_width - 1`. In each thread block, there are `w * w * Image_channels * sizeof(float)` bytes of memory read.

There are `gridx = (imageWidth - 1) / TILE_WIDTH + 1` number of thread blocks on x axis and `gridy = (imageHeight - 1) / TILE_WIDTH + 1` number of thread blocks on y axis.

There are total of `gridx * gridy` thread blocks for the image.

There are total of `gridx * gridy * w * w * Image_channels * sizeof(float)` bytes of global memory read for the image.

Q3

As we are implementing "output tiling method", all thread writes to global memory unless the thread is outside of image.

Each thread only writes to the global memory when it needs write a final output, which is a single float value.

Thus, the total global memory writes of the kernel for the image is `imageWidth * imageHeight * sizeof(float)` bytes.

Q4

Allocating GPU device memory, copying the data to device memory, retriving data from device memory to host, and freeing the GPU memory all counts as an overhead.

The memory allocation, copy, and free times are dominated by the size of the image, which is `imageWidth * imageHeight`. The overhead runtime grows linearly to the size of the image.

Q5

The shared memory usage will increase as the mask size increases along with the global memory reads. There will be more floating point operation by the square of the increase of the mask size. There is no change with the global memory write as the image size does not change.

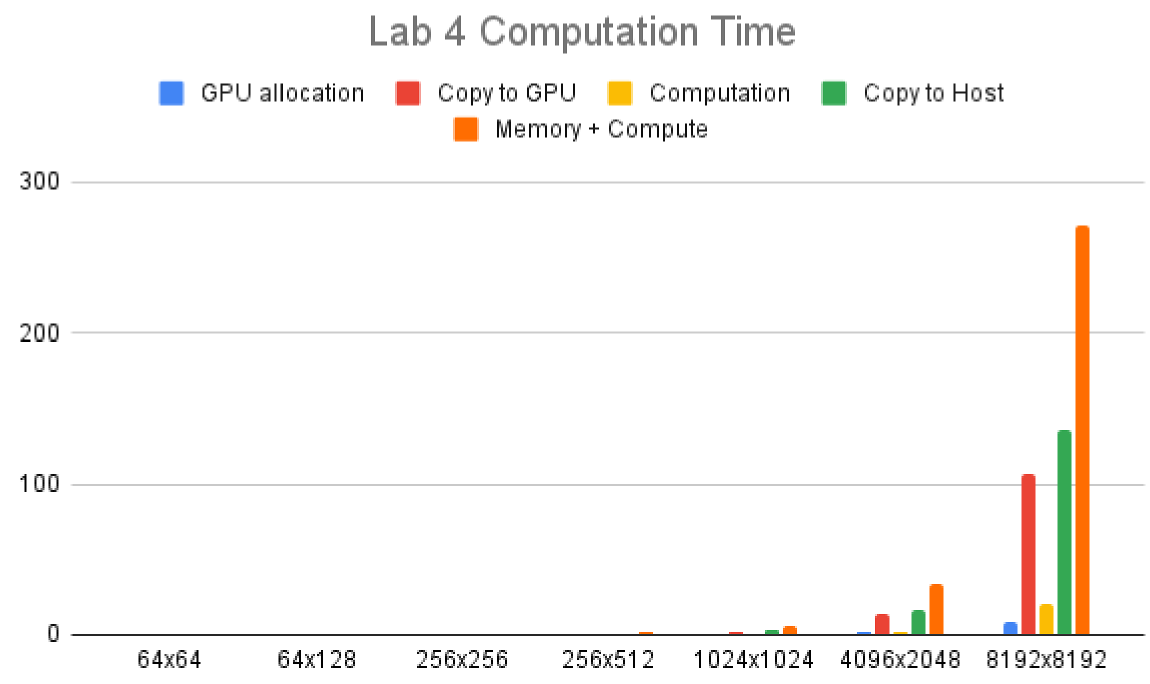
The kernel will be spending most of its runtime copying the halo cells required for the convolution when the mask size increases to 1024.

The shared memory will hit its limit as "each" thread block requires at least 4MB (4 * 1024 * 1024 bytes, which is just the mask dimension) of shared memory, which hinders the efficient parallel computation due to a lack of shared memory resource. The thread blocks may not be sufficiently allocated to SM due to the shared memory limit.

Q7

The kernel was run on RTX 3060ti with 8GB vram on Windows 11 WSL2 ubuntu 22.04

The runtime graph is as follows:



The raw data table is as follows:

	GPU allocation	Copy to GPU	Computation	Copy to Host	Memory + Compute
64x64	0.308276	0.374956	0.048069	0.061519	0.85078
64x128	0.255407	0.263857	0.044749	0.074269	0.697082
256x256	0.255779	0.290187	0.062469	0.284727	0.960211
256x512	0.458965	0.349187	0.08043	0.535534	1.48657
1024x1024	0.708622	1.66213	0.345206	2.98244	5.85018
4096x2048	1.75634	13.4847	2.39836	16.0602	33.789
8192x8192	8.4046	106.678	20.3233	136.14	271.664